

TIE-02306

Introduction to Software Engineering

5 credit units

06-swsys-ItSE2019-v4

Course contents (plan)

1. Course basics, intro
2. Sw Eng in general, overview
3. Requirements
4. **Basic UML Diagrams ("Class", Use Case, Navigation)**
5. UML diagrams, in more detail
6. **Different software systems**
7. Life Cycle models
8. Quality and Testing
9. Project work
10. Project management
11. Open source, APIs, IPR
12. Embedded systems
13. Recap

6. Different kind of software systems

- Different kind of software systems (by architecture)
 - stand-alone
 - client-server
 - mobile
 - web
 - real-time
 - reactive.
- SaaS etc.
- business, technical/engineering/scientific,...
- mission-critical / safety-critical systems
- distributed
- system software vs. application software
- architecture

Current at course (w 41)

- we have eight project groups left
- **WE6** groups this week are on WED and THU
- after those we think next week WE6 groups
- **to WE6 BYOC (bring your own computers); Dia tool**
- **Invite tunitensu to your group's Trello board**
- **Registration for EXAM 1/3 (weeks 41-43)**
- **1st phase return/delivery 13.10. at Moodle (opens 06.10.)**
- **Check that you can log into PRP system**

Backlog items with deadline

- **09.09.2019** at 23:59 Group forming (Moodle)
- **15.09.2019** at 23:59 Trello creation (Trello)
- **13.10.2019** at 23:59 Phase 1 documentation (Moodle)
- **13.10.2019** at 23:59 Phase 1 presentation slides (PRP-tool)
- **Week 43** Phase 1 presentations (Physical realm)
- **03.11.2019** at 23:59 Phase 1 peer feedback (PRP-tool)
- **17.11.2019** at 23:59 Phase 2 documentation (Moodle)
- **17.11.2019** at 23:59 Phase 2 presentation slides (PRP-tool)
- **Week 47** Phase 2 presentation (Physical realm)
- **01.12.2019** at 23:59 Phase 2 peer feedback (PRP-tool)
- **08.12.2019** at 23:59 Final delivery of project documentation (Moodle)
- **15.12.2019** at 23:59 Final peer feedback and self assessments (PRP-tool).

08.10.2019

TIE-02306

11

Weekly exercise attendees

	w36 WE1	w37 WE2	w38 WE3	w39 WE4	w40 WE5	w41 WE6	w44 WE7	w45 WE8	w46 WE9	w48 WE10
WED	0	14	9	5	8					
THU	21	13	14	17	16					

We will continue two Weekly Exercise groups, as long as the number of attendees are reasonable.

- **Software systems**
- **Architectures**

(it is difficult, if not impossible, to define one clear exact definition for those...)

Several classifications of sw systems

Basis of **application** [geeksforgeeks.org]:

- system software / network or web application / embedded / business / entertainment or gaming / artificial intelligence / scientific / document management / utilities.

Basis of **copyright** [geeksforgeeks.org]:

- commercial / shareware / freeware / public domain.

Classification of **software** [ecomputernotes.com]

- system sw / real-time sw / business sw / engineering and scientific / artificial intelligence / web-based / personal computer.

Basis of **deployment** platform...

- PC, mobile, client-server, cloud, hosted (SaaS),...

Software system types, 1

- **System software**
 - e.g. operating systems, tools
- **Mobile application**
 - e.g. GSM, tablet
- **Organisational information systems**
 - application programs
- **Network services**
 - e.g. internet, www.

FI: sovelma = applet (www-selaimessa toimiva ohjelma)

FI: sovellus = application.

Software system types, 2

- **Embedded systems**
 - inside a machine or device
 - e.g. lift/elevator control system
- **Real-time systems**
 - software must react immediately
 - e.g. fuel and breaks control in cars
- **Reactive systems**
 - operate continuously
 - e.g. GSM switch/exchange centre.

Almost all devices nowadays contain software; refrigerator, door lock, coffee machine, lamp,...

Just another software classification

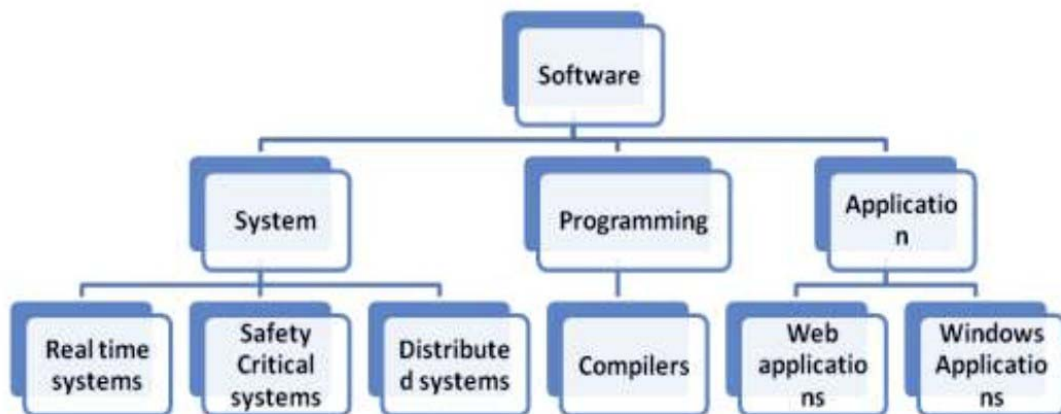
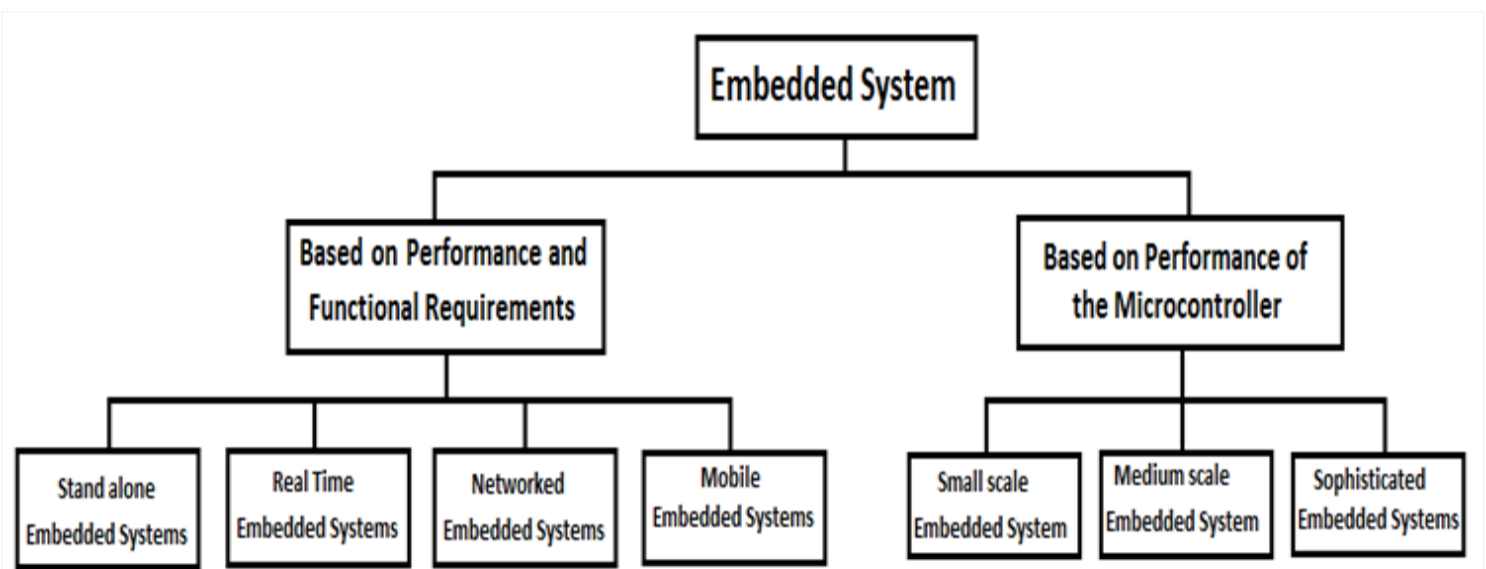


Figure 1. Software Classification

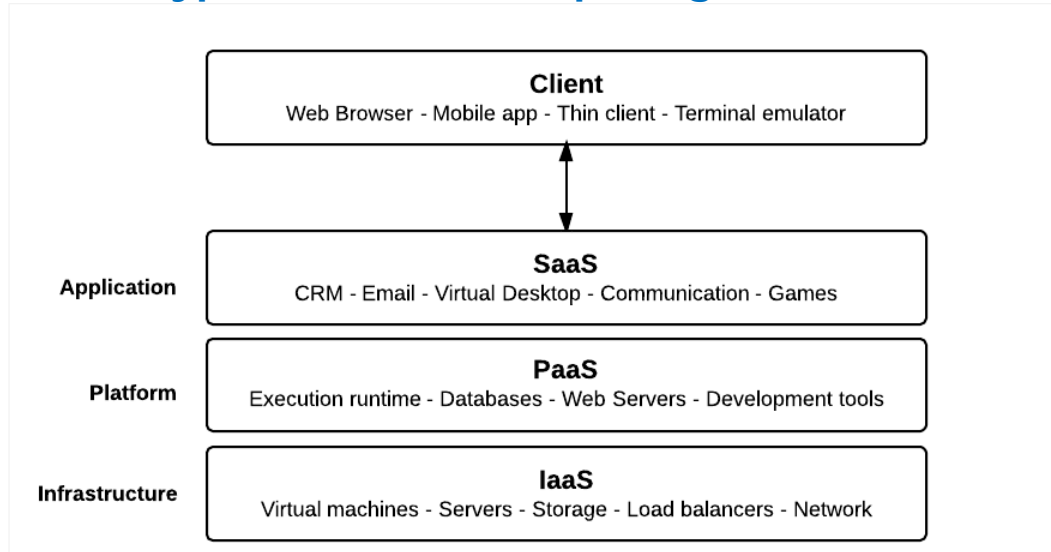
[https://www.researchgate.net/figure/Software-Classification_fig5_288645078]

One classification of embedded systems



[<https://er.yuvayana.org/classification-of-embedded-system-with-details/>]

Different Types of Cloud Computing Service Models



[<https://www.bluepiit.com/blog/different-types-of-cloud-computing-service-models/>]

Software Systems Taxonomy

(As found in Alleman's "Agile Project Management Methods for IT Projects")

Software Type	Attributes
Management Information Systems	Software that an enterprise uses to support business and administrative operations.
Outsourced Systems	Software projects built for client organizations.
Systems Software	Software that controls a physical device such as a computer or a telephone switch.
Commercial Software	Software applications that are marketed to hundreds or even millions of clients.
Military Software	Software produced for the uniformed services.
End User Software	Small applications written for personal use.
Web Application and e- Projects	Small, medium, and large scale projects with legacy system integration, transaction processing, multimedia delivery, and web browser based user interfaces.

3.3829 , software product

1. set of computer programs, procedures, and possibly associated documentation and data [ISO/IEC 12207:2008 Systems and software engineering — Software life cycle processes, 4.42; ISO/IEC 25000:2014 Systems and software Engineering — Systems and software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE 4.31; ISO/IEC TS 24748-1:2016 Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management, 2.48]

2. any of the individual items in (1)

3. complete set of software designed for delivery to a software consumer or end-user, which can include computer programs, procedures and associated documentation and data. [ISO/IEC 19770-5:2015 Information technology — IT asset management — Part 5: Overview and vocabulary, 3.46]

4. set of computer programs, procedures, database- and other data structure descriptions and associated documentation [ISO/IEC 16350-2015 Information technology — Systems and software engineering — Application management, 4.33]

cf. software package

Note 1 to entry: A software product can be designated for delivery, an integral part of another product, or used in development. Software products vary from large customized application software for one customer to standard software packages that are sold off the shelf to millions of customers.

3.216 , architecture

1. [system] **fundamental concepts or properties of a system** in its environment embodied in its elements, relationships, and in the principles of its design and evolution [ISO/IEC TS 24748-1:2016 Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management, 2.6; ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes, 4.1.5; ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description, 3.2]

2. set of rules to define the structure of a system and the interrelationships between its parts [ISO/IEC 10746-2:2009 Information technology — Open Distributed Processing — Reference Model: Foundations, 6.6]

cf. component, module, subprogram, routine.

3.3870 , software/system element

1. element that defines and prescribes what a software or system is composed of [ISO/IEC TR 18018:2010 Information technology — Systems and software engineering — Guide for configuration management tool capabilities, 3.12]

cf. software element

EXAMPLE: requirements, design, code, test cases, and version number.

3.4090 , system

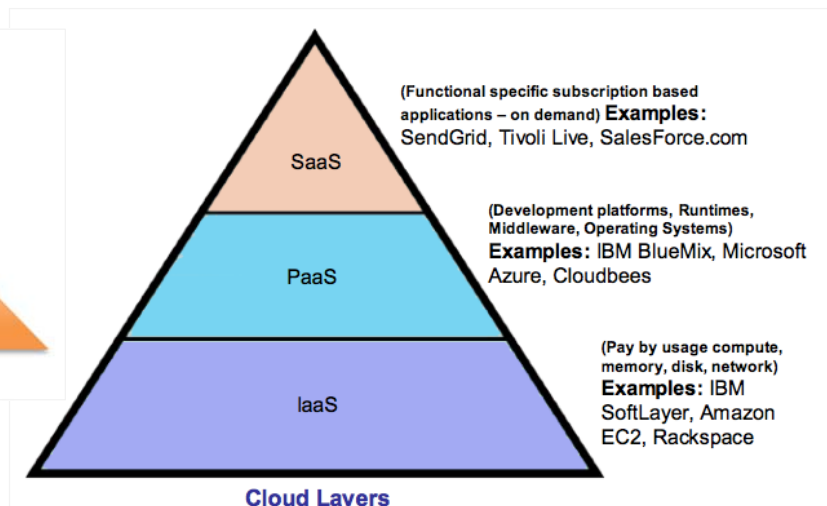
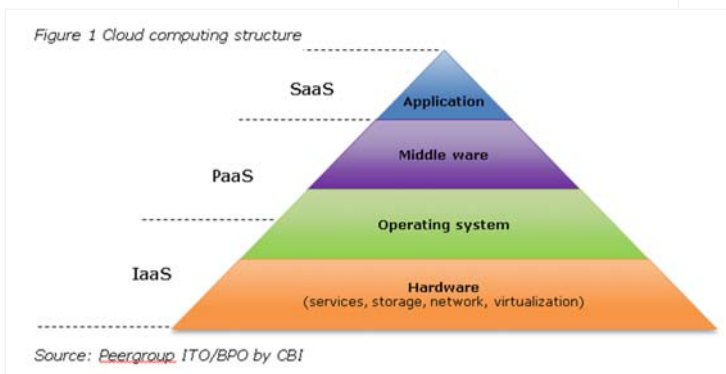
1. combination of interacting elements organized to achieve one or more stated purposes [ISO/IEC/IEEE 15939:2017 Systems and software engineering — Measurement process, 3.38; ISO/IEC 25000:2014 Systems and software Engineering — Systems and software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE, 4.38; ISO/IEC TR 90005:2008 Systems engineering — Guidelines for the application of ISO 9001 to system life cycle processes, 2.1; ISO/IEC TS 24748-1:2016 Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management, 2.53; ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes, 4.1.46]
2. product of an acquisition process that is delivered to the user [IEEE 15288.2:2014 IEEE Standard for Technical Reviews and Audits on Defense Programs, 3.1]
3. something of interest as a whole or as comprised of parts [ISO/IEC 10746-2:2009 Information technology — Open Distributed Processing — Reference Model: Foundations, 6.5]
4. interacting combination of elements to accomplish a defined objective [ISO/IEC TR 19759:2016 Software Engineering — Guide to the Software Engineering Body of Knowledge (SWEBOK), 1.1.6]
5. set of interrelated or interacting elements [ISO/IEC TR 90005:2008 Systems engineering — Guidelines for the application of ISO 9001 to system life cycle processes, 2.2]

Note 1 to entry: A system is sometimes considered as a product or as the services it provides. In practice, the interpretation of its meaning is frequently clarified by the use of an associative noun, e.g., aircraft system. Alternatively, the word 'system' can be replaced by a context-dependent synonym, e.g., aircraft, though this obscures the system perspective. A complete system includes all of the associated equipment, facilities, material, computer programs, firmware, technical documentation, services, and personnel required for operations and support to the degree necessary for self-sufficient use in its intended environment.

TAU/TUNI * TIE-02306 Introduction to Sw Eng

08.10.2019

23



SaaS = sw as a service
PaaS = platform as a service
IaaS = infrastructure as a service.

[www.ibm.com/blogs/cloud-computing/]

Cloud computing

There are several types of cloud computing. Providers offer infrastructure, platforms or software as a web-based service. Instead of purchasing these components, clients subscribe to them as a variable-cost service. The service provider owns the components and is responsible for housing, running and maintaining them.

Infrastructure as a Service (IaaS)

- the service provider offers computer infrastructure components as a web-based service
- for example servers, storage, data centre space and network equipment

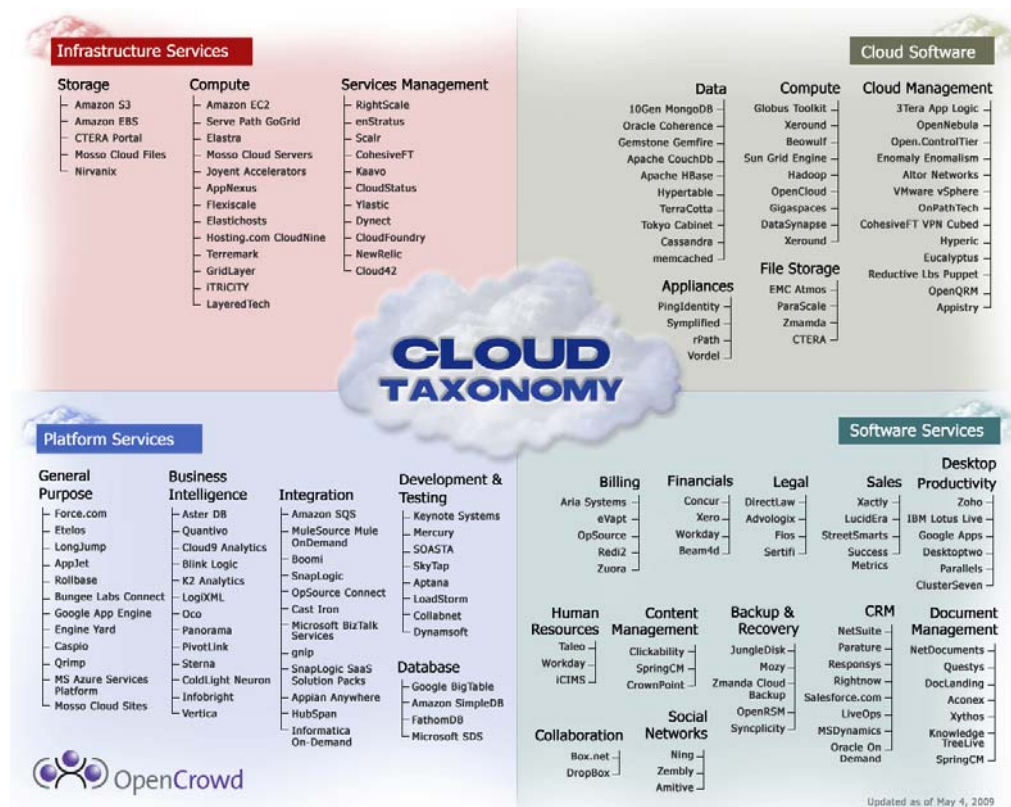
Platform as a Service (PaaS)

- the service provider offers a computing platform as a web-based service
- this typically includes an operating system, programming language, execution environment, database and web server

Software as a Service (SaaS)

- the service provider offers software applications as a web-based service
- for example horizontal business applications like Customer Relationship Management (CRM), Enterprise Resource Planning (ERP), Content Management (CM), Human Resource Management (HRM) or finance and accounting.

[<https://www.cbi.eu/market-information/outsourcing-bpo-ito/cloud-computing/europe/>]



Architecture Framework

Definition of Software Architecture... it depends...

Carnegie Mellon University
Software Engineering Institute



What is your definition of software architecture?

WHAT IS YOUR DEFINITION OF SOFTWARE ARCHITECTURE?

The SEI has compiled a list of modern, classic, and bibliographic definitions of software architecture.

Modern definitions are definitions from Software Architecture in Practice and from *ANSI/HFPP-98*

emphasizes the plurality of structures present in every software system. These structures, carefully chosen and designed by the architect, are the key to achieving and reasoning about the system's design goals. And those structures are the key to understanding the architecture. Therefore, they are the focus of our approach to

[<https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513807>], 2017

Architecture vs. framework, 1

Framework is an implementation of an architecture.

- - -

An **architecture** is the the abstract design concept of an application. Basically, a structure **of the** moving parts and how they're connected. A **framework** is a pre-built general or special purpose **architecture** that's designed to be extended.

- - -

If an architecture is the design of a structure, a framework is the architecture of a foundation. Frameworks are specifically designed to be built on or extended.

- - -

[softwareengineering.stackexchange.com]

Architecture vs. framework, 2

Generally speaking, **architecture** is an abstract plan that can include design patterns, modules, and their interactions. **Frameworks** are *architected* "physical" structures on which you build your application.

Your **architecture may incorporate multiple frameworks**. And multiple layers of framework. At each layer, the "architecture" is the often-documented "thinking" from which the layer is built. The frameworks are the "physical" components used to build it.

- - -

In software, a Framework is a software module or set of modules that supports a generic programming concept by abstracting common functionality (code in a software sense) into a reusable format.

An Architecture is an assembly of systems that solves business needs.

- - -

[softwareengineering.stackexchange.com]

Architectural views [Sommerville]

It is impossible to represent all relevant information about a system's architecture in a single diagram, as a graphical model can only show one view or perspective of the system. It might show how a system is decomposed into modules, how the runtime processes interact, or the different ways in which system components are distributed across a network. Because all of these are useful at different times, for both design and documentation, you usually need to present multiple views of the software architecture.

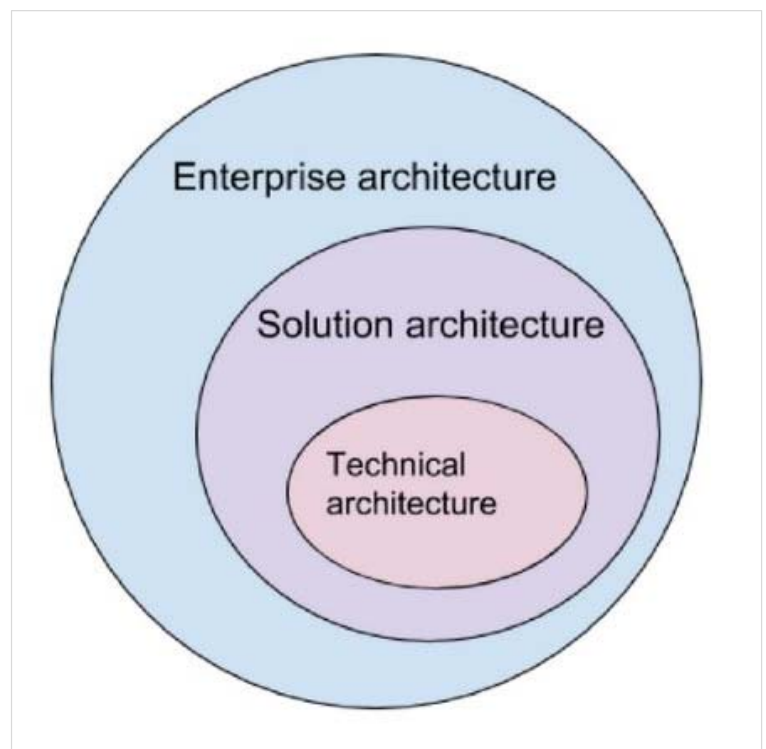
There are different opinions as to what views are required.

1. A **logical** view, which shows the key abstractions in the system as objects or object classes. It should be possible to relate the system requirements to entities in this logical view.
2. A **process** view, which shows how, at runtime, the system is composed of interacting processes. This view is useful for making judgments about non-functional system characteristics such as performance and availability.
3. A **development** view, which shows how the software is decomposed for development; that is, it shows the breakdown of the software into components that are implemented by a single developer or development team. This view is useful for software managers and programmers.
4. A **physical** view, which shows the system hardware and how software components are distributed across the processors in the system. This view is useful for systems engineers planning a system deployment.
5. A **Conceptual** view.

Enterprise architect focuses on building complex enterprise ecosystems and solves high-level strategic issues. Enterprise architecture defines strategic directions of the business architecture, which then leads to an understanding of what technology facilities are needed to support that architecture.

Solution architecture (SA) is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions.

Technical architect is mainly in charge of engineering problems and software architecture.



[<https://www.altexsoft.com/blog/engineering/solution-architect-role/>]

Just another classification of architecture layers

Enterprise architecture (FI: kokonaisarkkitehtuuri, JHS 179-2017)

Software architecture (FI: ohjelmistoarkkitehtuuri)

Systems architecture (FI: järjestelmäarkkitehtuuri).

“An architecture is the result of a set of business and technical decisions.”

About architecture, 1

Having a **software architecture** is important to the successful development of a software system.

Software systems are constructed to satisfy organizations' business goals. The **architecture is a bridge between** those (often abstract) **business goals and the final** (concrete) **resulting system**. While the path from abstract goals to concrete systems can be complex, the good news is that software architectures can be designed, analyzed, documented, and implemented using known techniques that will support the achievement of these business and mission goals. The complexity can be tamed, made tractable.

[Len Bass et al.: Software Architecture in Practice, 3rd ed., 2013]

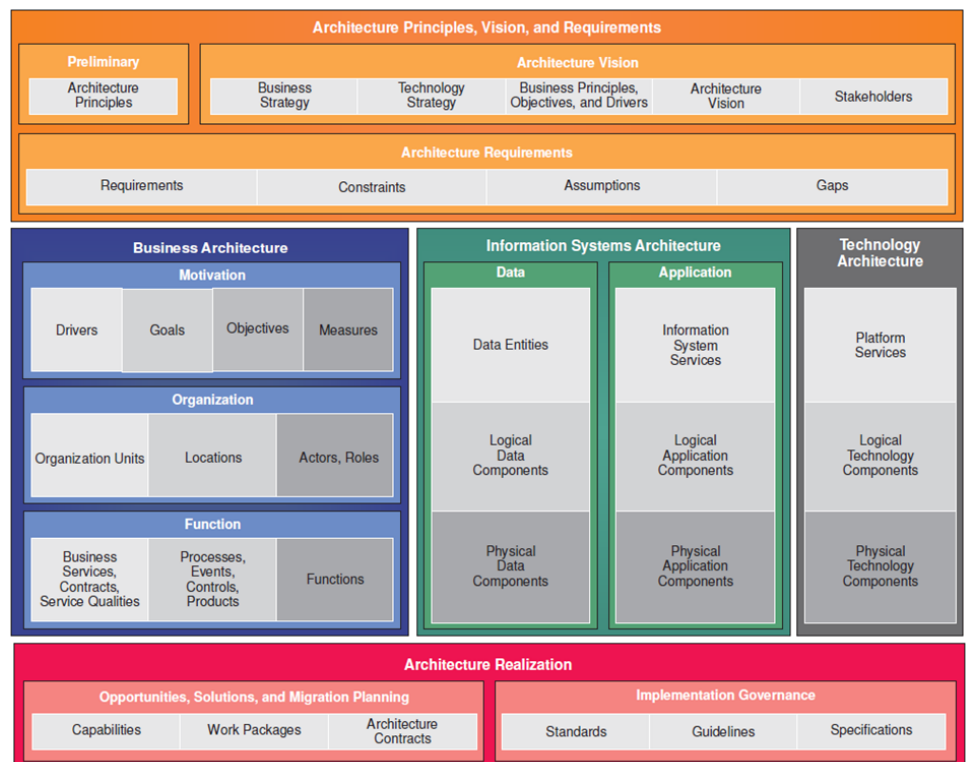
About architecture, 2

Architecture is defined by the recommended practice as the *fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution*. This definition is intended to encompass a variety of uses of the term architecture by recognizing their underlying common elements. Principal among these is the need to understand and control those elements of system design that capture the system's utility, cost, and risk. In some cases, these elements are the physical components of the system and their relationships. In other cases, these elements are not physical, but instead, logical components. In still other cases, these elements are enduring principles or patterns that create enduring organizational structures.

[ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems]

TOGAF =
The Open Group
Architecture Framework

(FI: kokonaisarkkitehtuuri-
viitekehys)



[<https://guozspace.com/2013/04/28/togaf-9-1-content-metamodel-overview/>]

Roles

Difference between IT Architect Roles

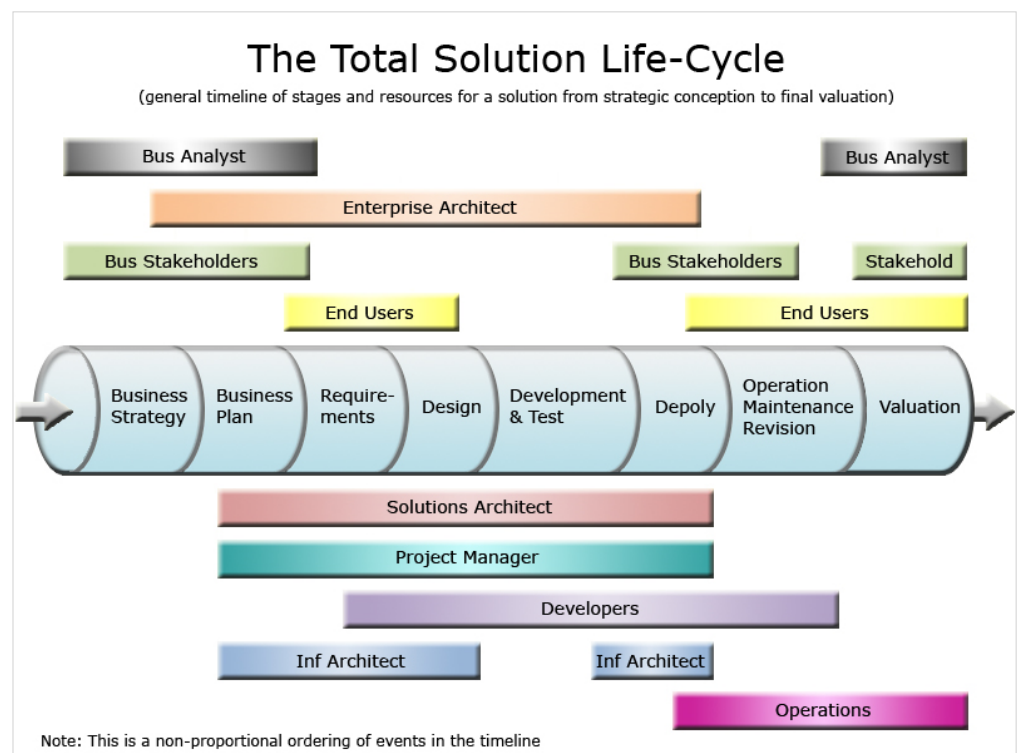
Enterprise Architect	Solutions Architect	Technical Architect
<ul style="list-style-type: none"> Enterprise Architect (EA) is a planning role that is responsible for identifying the future state of an organization's IT environment and engage wherever and whenever necessary to help guide project teams to deliver toward it. An EA would ensure IT investments are aligned with business strategy. Responsible for strategic thinking, roadmaps, principles and governance of the entire enterprise. Enterprise Architect defines which problem need a solution 	<ul style="list-style-type: none"> Solutions Architect (SA) focuses on delivery of a particular solution. The SA is responsible for implementing a strategic IT program within the framework laid down by the enterprise architecture (EA) team. Solutions Architect (SA) is assigned to ensure technical integrity and consistency of the solution on every stage of its life-cycle Solutions Architect translates a problem to a solution 	<ul style="list-style-type: none"> Technical Architect is usually a technology specialist in a particular technology or group of interrelated technologies. Job titles vary for this role and they may also include Infrastructure Architect, Domain Architect, Application Architect (Java Architect, .Net Architect), Network Architect, Security Architect . Technical Architect works within a solution

[<https://image.slidesharecdn.com/ss-180127053749/95/solution-architecture-8-638.jpg?cb=1517031523>]

8

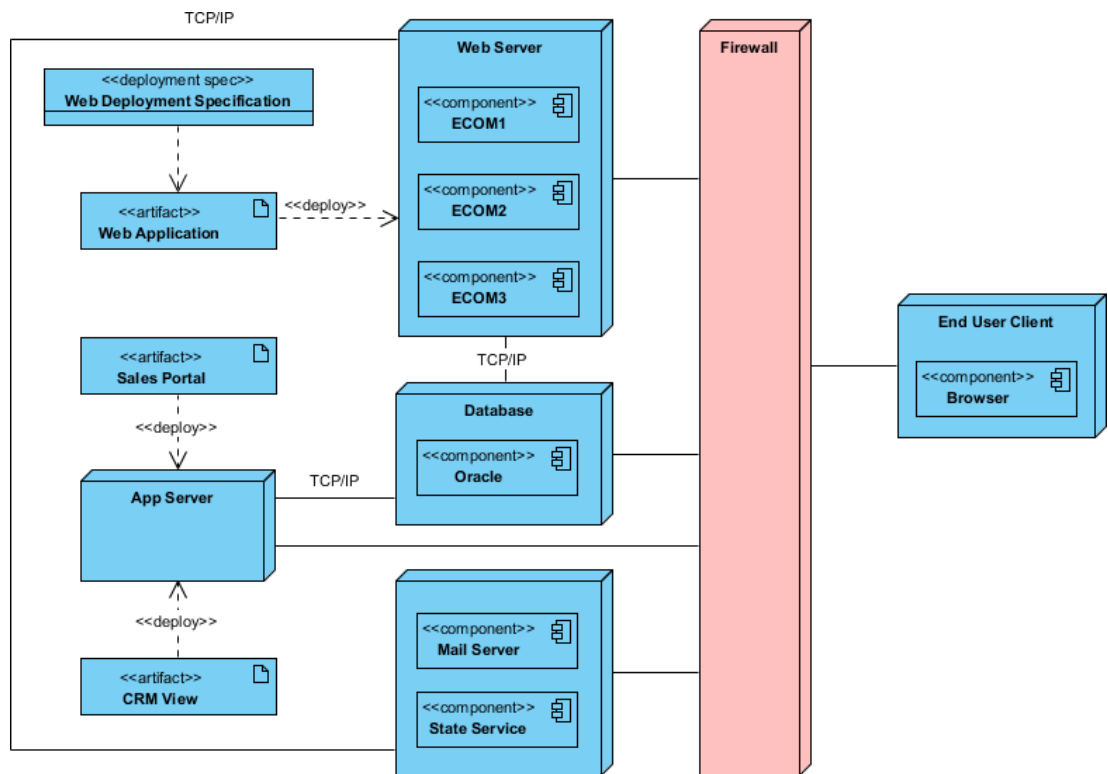
<http://blog.danovich.com.au/2012/03/24/differences-between-it-architect-roles/> | <http://sysarchitect.com/2011/09/19/differences-between-architecture-roles/> | <http://sysarchitect.com/2011/09/19/differences-between-architecture-roles/>

The total solution life cycle includes the business strategy and business planning activities that precede the software development life cycle (SDLC), as well as the deployment and ongoing operations that follow. We examine the tools used to create business strategies, develop and implement solutions, and evaluate solution effectiveness. A solutions architect that understands the total solution life cycle can make better decisions than a peer who understands only the SDLC.



[[https://docs.microsoft.com/en-us/previous-versions/bb756611\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/bb756611(v=msdn.10)?redirectedfrom=MSDN)]

APIs
(interfaces,
connections)
are taken
into account,
i.e. planned
at design
phase.



[<https://www.archimetric.com/what-is-deployment-diagram/>]

Some architecture standards

- ISO/IEC/ IEEE 42010:2011

Systems and software engineering — Architecture description

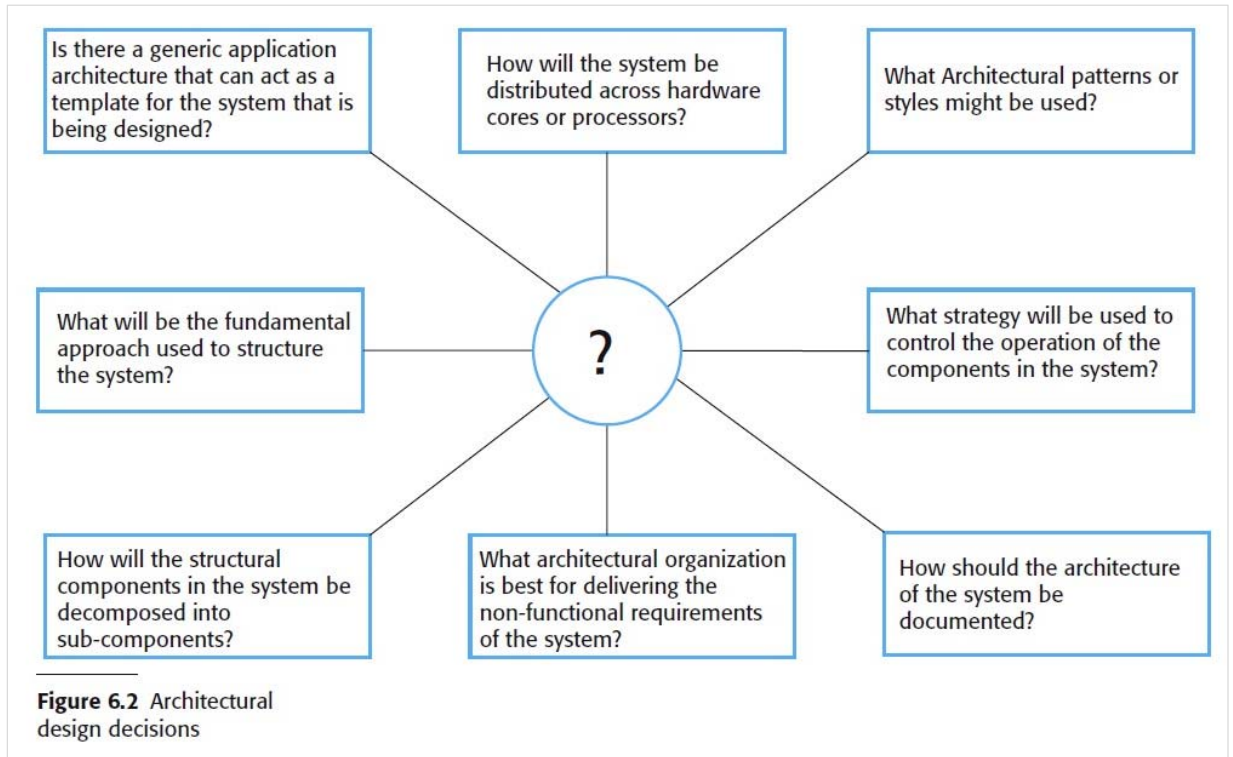
- IEEE 42020:2019

ISO/IEC/IEEE International Standard - Software, systems and enterprise -- Architecture processes

- ISO/IEC/IEEE 42030:2019

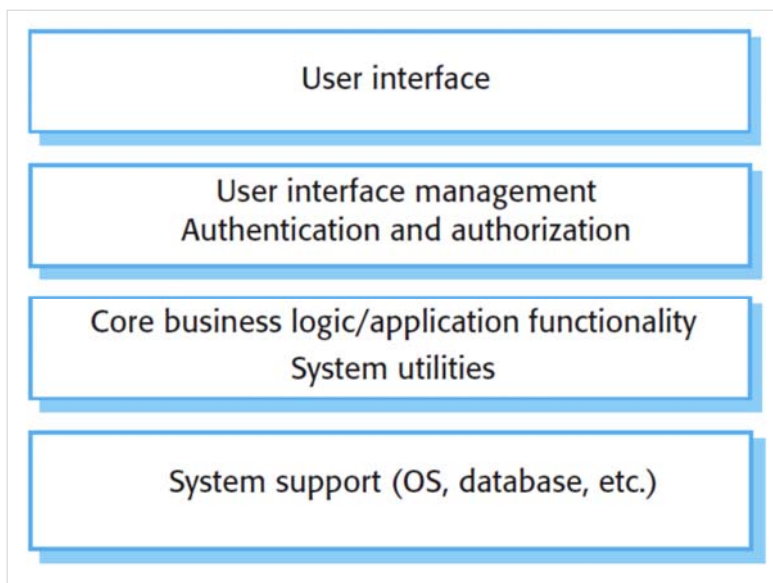
Software, systems and enterprise — Architecture evaluation framework

“An architecture is what is fundamental to a system — not necessarily everything about a system, but the essentials.”



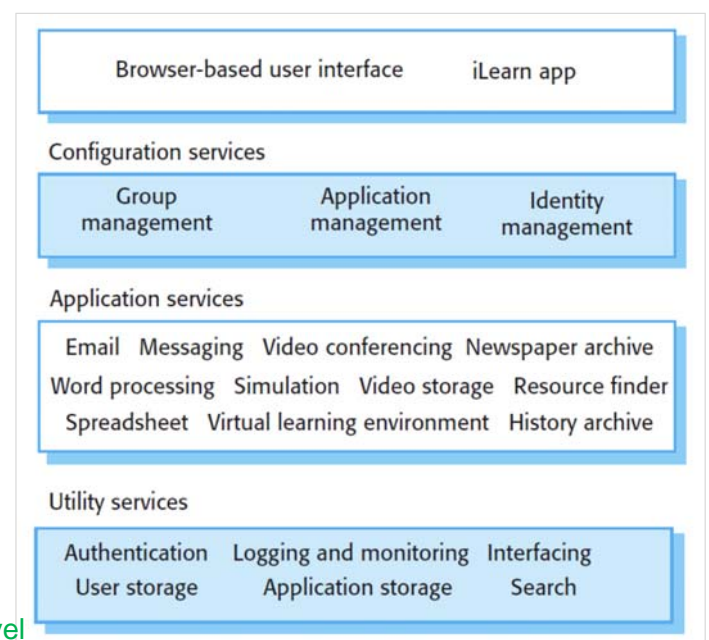
One example of architecture layers [Sommerville, 2016]

high level



Generic layered architecture.

low level



iLearn system architecture

Architecture matters [Sommerville, 2016]

A software architecture is a description of how a software system is organized. Properties of a system such as performance, security, and availability are influenced by the architecture used.

- Architectures may be documented from several different perspectives or views. Possible views include a **conceptual** view, a **logical** view, a **process** view, a **development** view, and a **physical** view.
- Architectural patterns are a means of reusing knowledge about generic system architectures. They describe the architecture, explain when it may be used, and point out its advantages and disadvantages.
- Commonly used Architectural patterns include model-view-controller, layered architecture, repository, client–server, and pipe and filter.
- Generic models of application systems architectures help us understand the operation of applications, compare applications of the same type, validate application system designs, and assess large-scale components for reuse.
- Language processing systems are used to translate texts from one language into another and to carry out the instructions specified in the input language. They include a translator and an abstract machine that executes the generated language.

Terminology, ISO/IEC/IEEE 24765:2017 standard

3.169 , application architecture

1. architecture including the architectural structure and rules (e.g. common rules and constraints) that constrains a specific member product within a product line [ISO/IEC 26550:2015 *Software and systems engineering — Reference model for product line engineering and management*, 3.1]

Note 1 to entry: The application architecture captures the high-level design of a specific member product of a product line.

3.210 , architecting

1. process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system's life cycle.

3.214 , architectural structure

1. physical or logical layout of the components of a system design and their internal and external connections.

2.2 Software Architecture Description

Stakeholders have different viewpoints and views

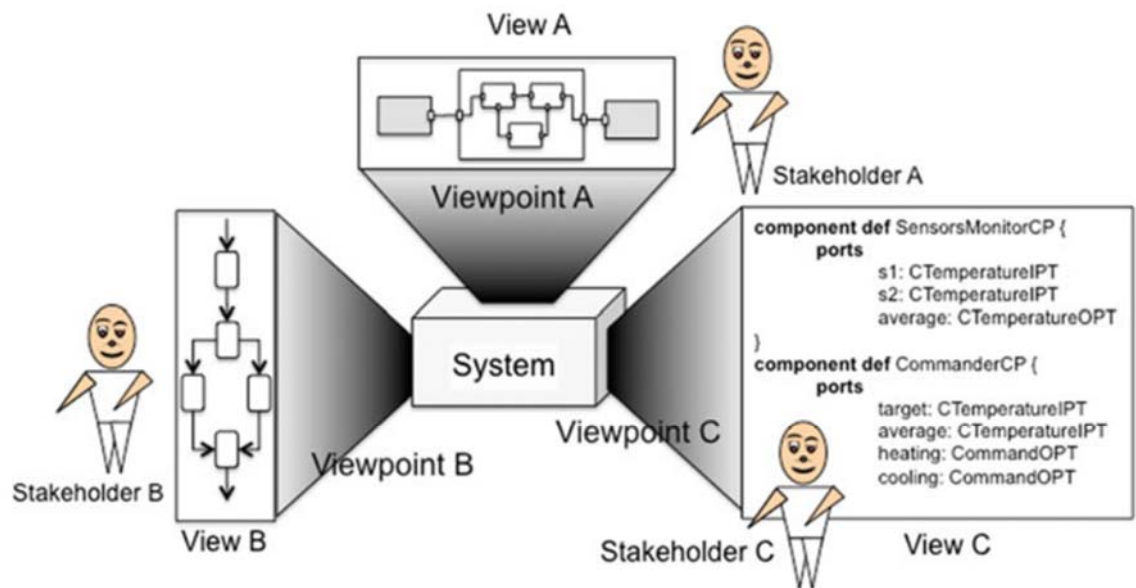


Fig. 2.5 Viewpoints and views

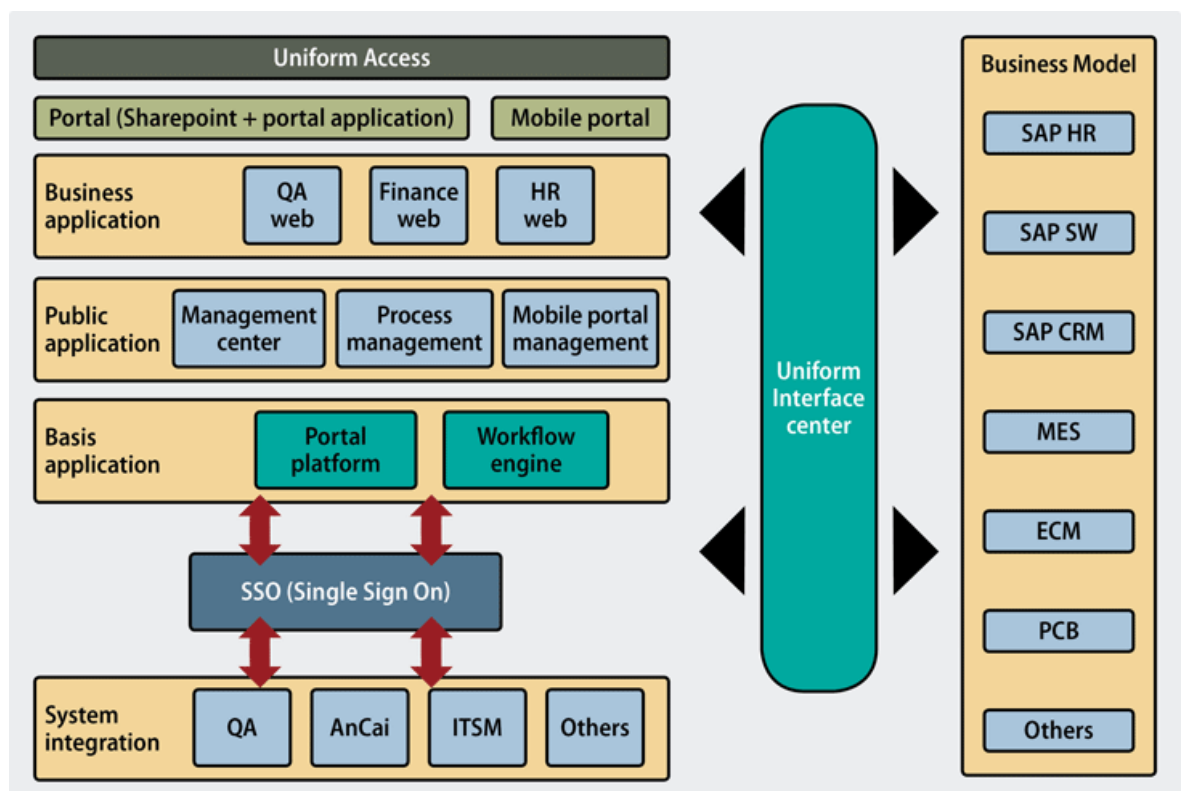
[Software Architecture in Action, 2016]

08.10.2019

TIE-02301

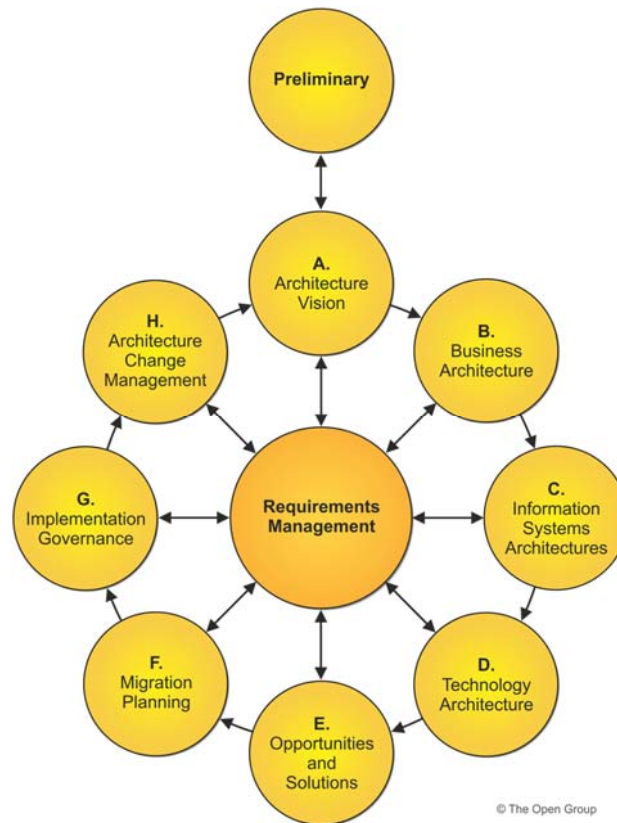
45

In a typical example of **enterprise architecture**, all elements are included in the overall schema, from business models to business applications to system integration workflow.



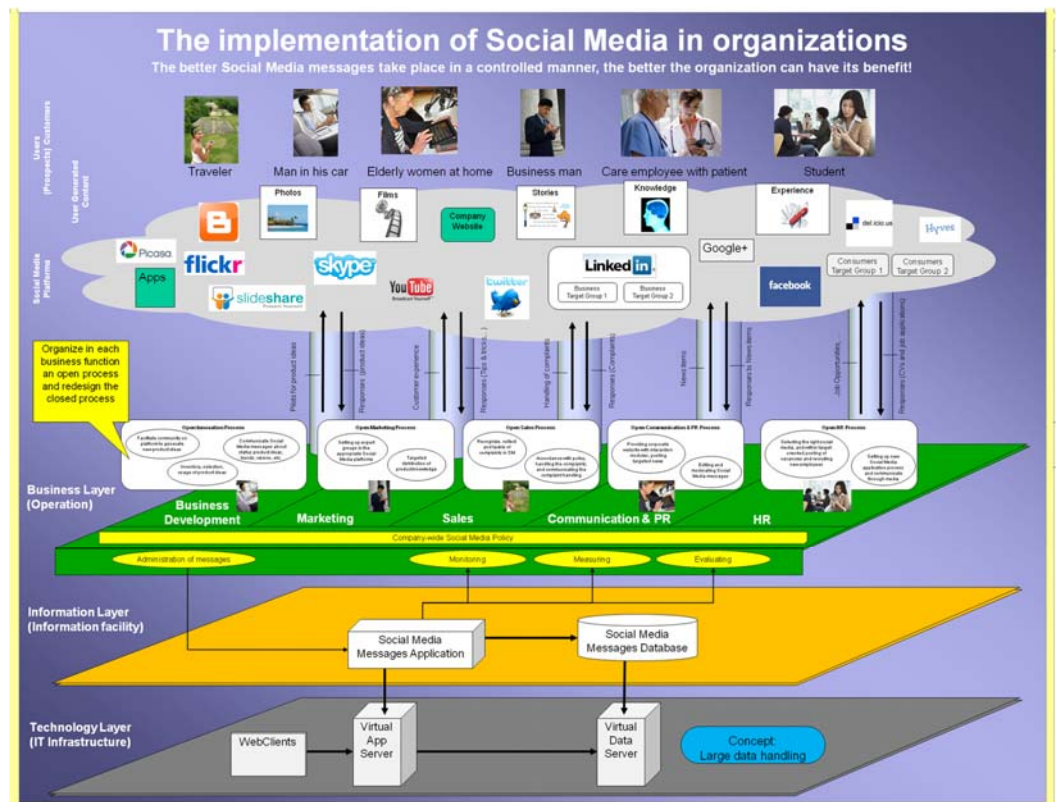
TOGAF (The Open Group
Architecture Framework)
9.2 architecture

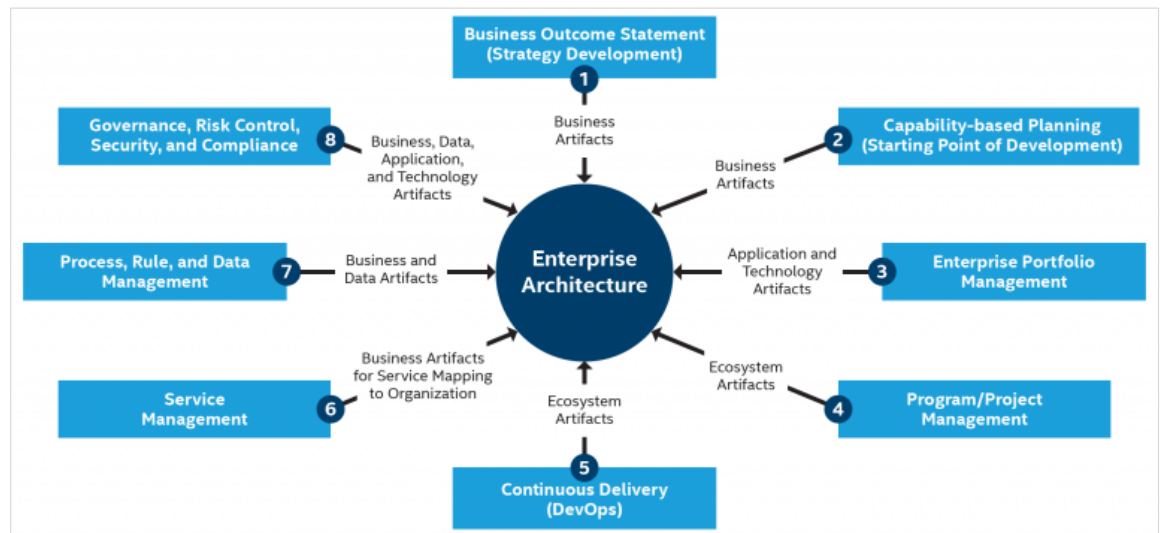
Planning **enterprise architecture** requires balancing a wide variety of governance, technology and management issues.



Architectural view;

"Social Media 2.0;
The Way It Works".





" Enterprise architects have a tough job. They have to think strategically but act tactically. A successful enterprise architect can sit down at the boardroom table and discuss where the business needs to go, then translate "business speak" into technical capabilities on the back end. The key to EA is to always **focus on business needs first**, then how those needs can be met by applying technology."

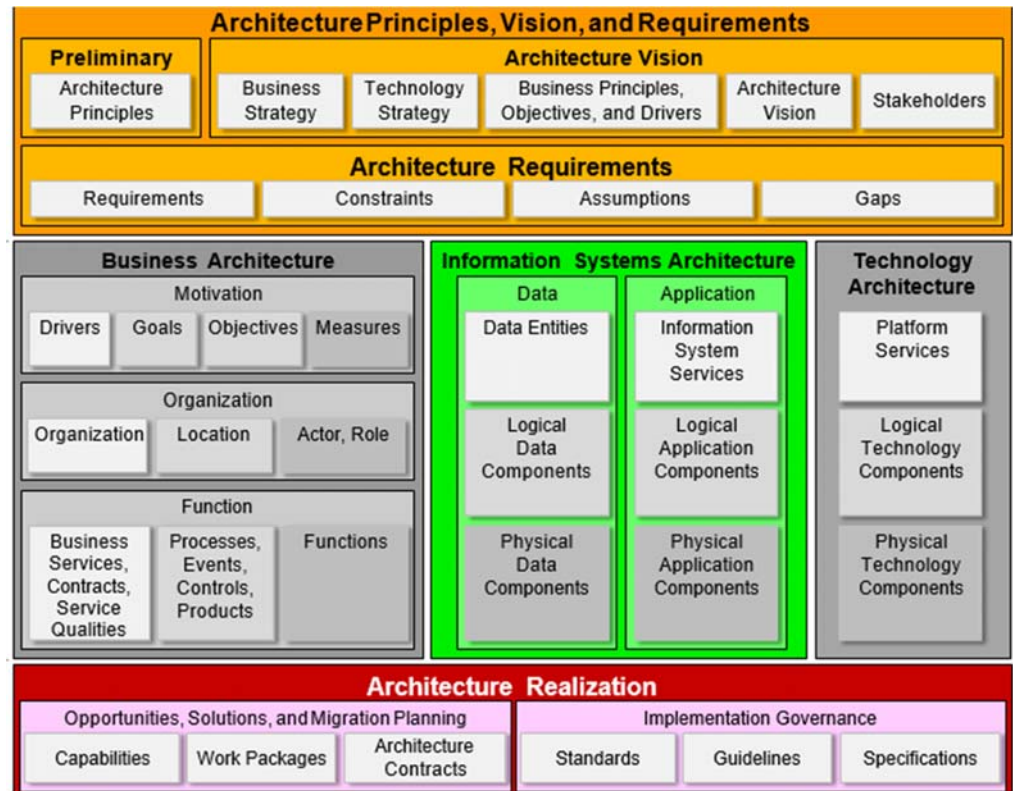
[<https://itpeernetwork.intel.com/enterprise-architecture/>]

The Scope of Focus in Zachman EA Framework

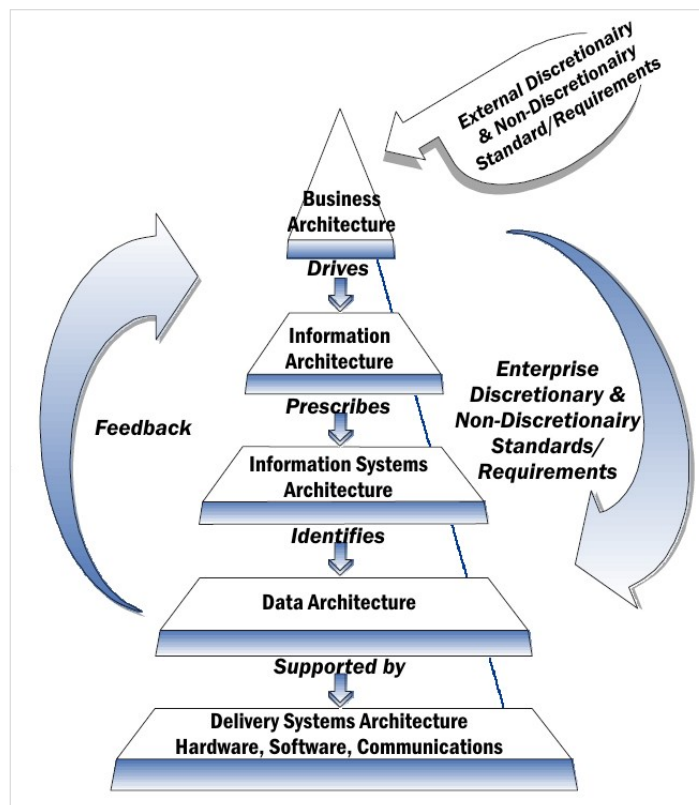
Views	What	How	Where	Who	When	Why
	Entity/Data	Function	Network	People	Time	Motivation
Business Environment (Planner)	Info/Data Architecture (scope of focus)	Enterprise Architecture (scope of focus)				
Capabilities, Functions, Services (Owner)						
Logical Systems (Designer)		Solution Architectures (scope of focus)				
Physical Systems (Builder)						
Technology References		Technology Architectures (scope of focus)				
Technology Details Specification						

Yan Zhao, Ph.D, ArchiTech Consulting LLC, 2011

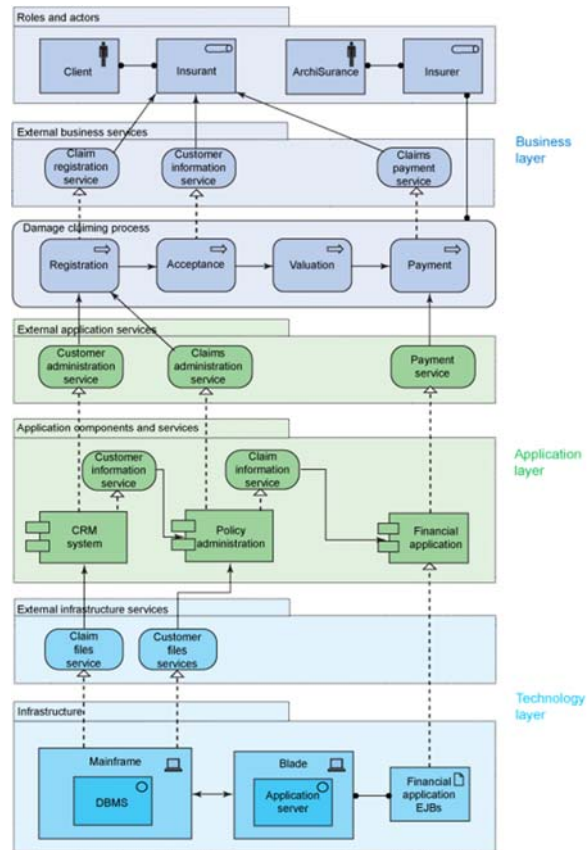
TOGAF 9



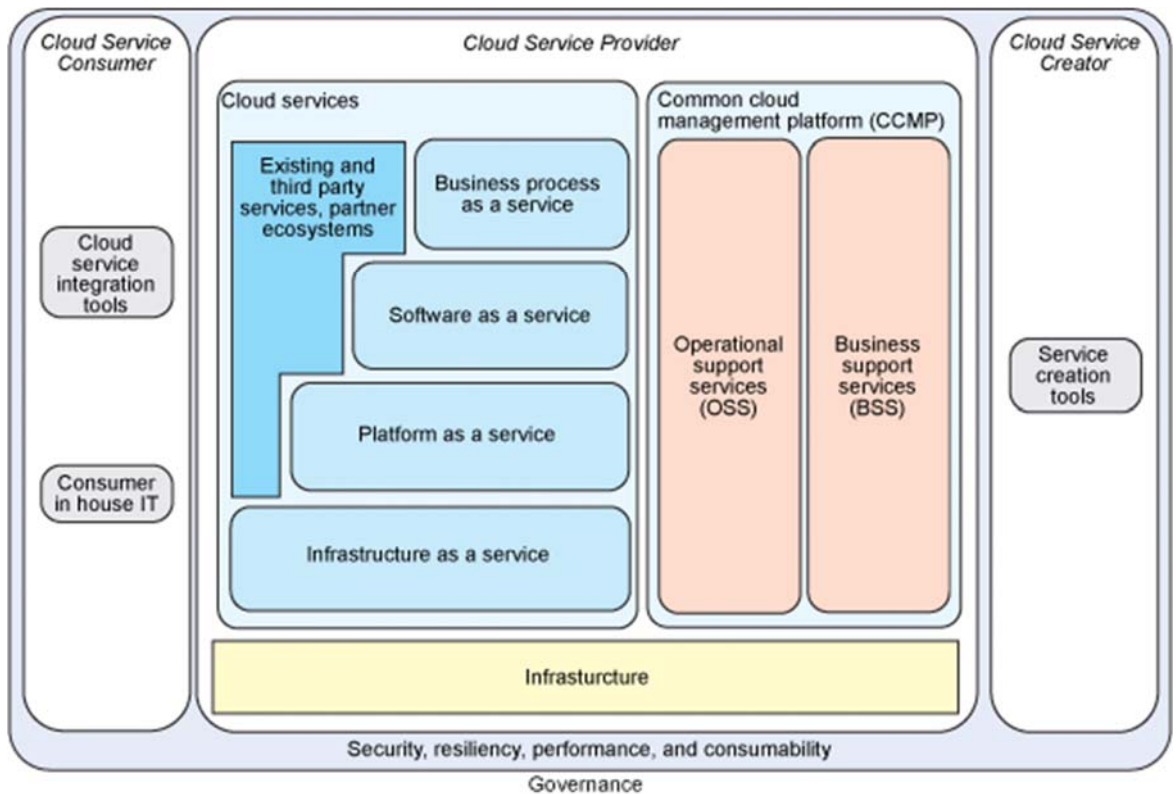
NIST (National Institute of Standards and Technology) Enterprise Architecture



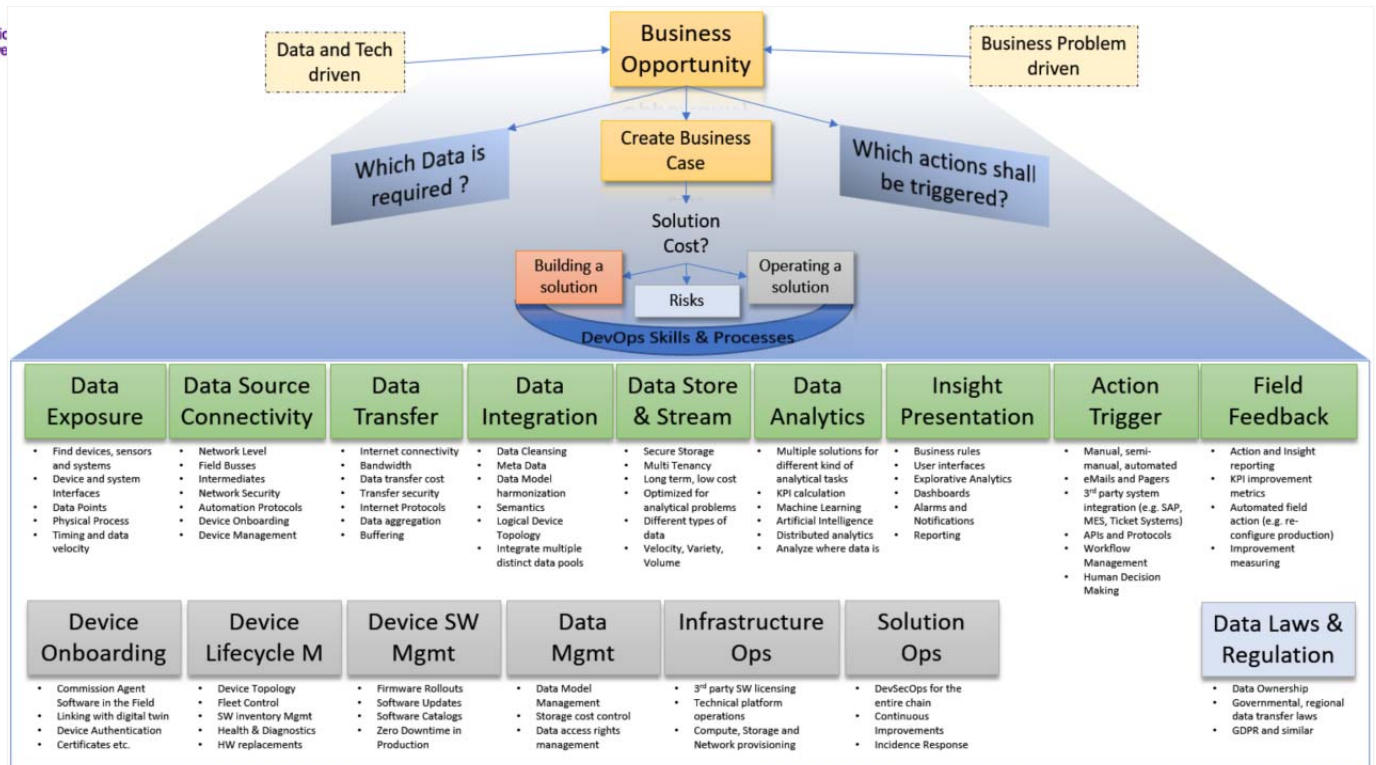
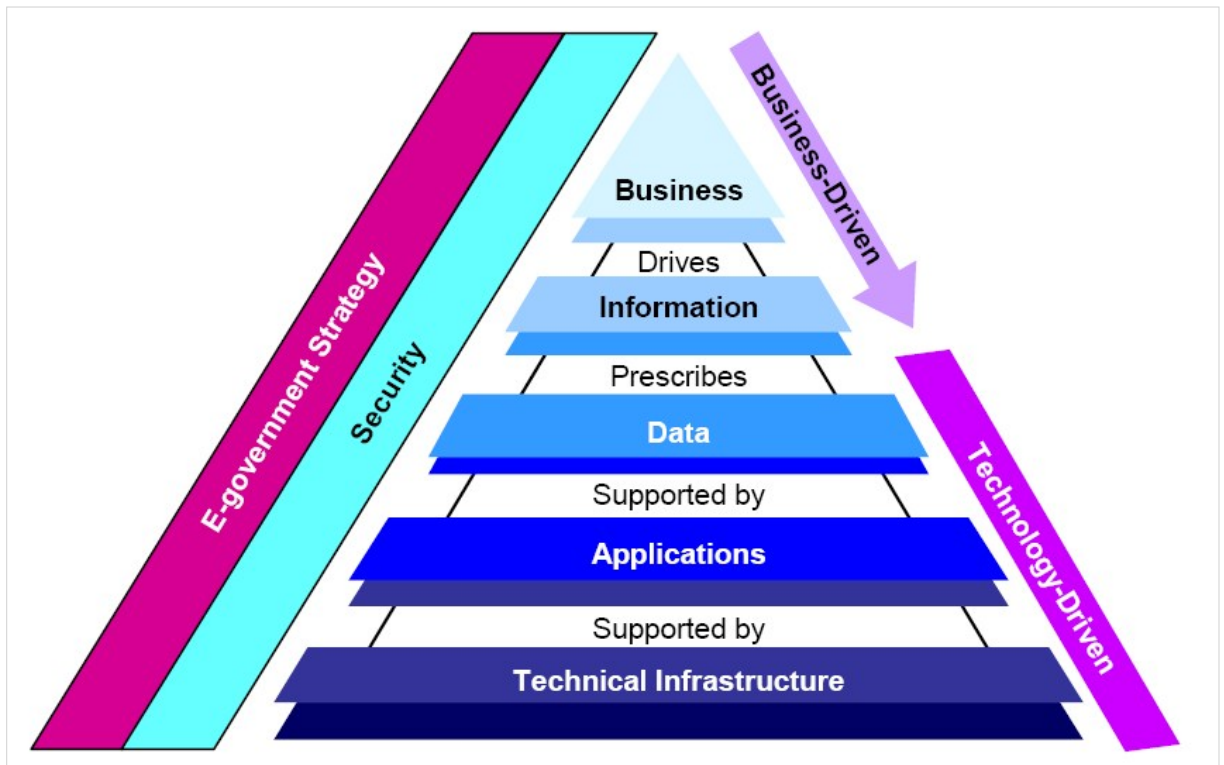
IBM architecture



IBM Cloud Architecture



FDIC
(Federal
Deposit
Insurance
Company)
enterprise
architecture
framework



[<https://ideal-digital.org/2019/04/17/a-taxonomy-of-industrial-iiot-solution-building/>]

Zachman's 6x6 matrix.





































The Zachman Framework is an enterprise ontology and is a fundamental structure for Enterprise Architecture which provides a formal and structured way of viewing and defining an enterprise.

Generic Classification Structure of Design Artifacts							
	What	How	Where	Who	When	Why	
Planner							Scope
Owner							Concepts
Designer							Logic
Builder							Physics
Implementer							Technology
Operator	THE ENTERPRISE						Product
	Material	Process	Geometry	Instructions.	Timing	Objectives	

Zachman...

	Data (What)	Function (How)	Network (Where)	People (Who)	Time (When)	Motivation (Why)
Objectives / Scope	List of things important to the enterprise	List of processes the enterprise performs	List of locations where the enterprise operates	List of organizational units	List of business events / cycles	List of business goals / strategies
Business Model	Entity relationship diagram (including m:m, n-ary, attributed relationships)	Business process model (physical data flow diagram)	Logistics network (nodes and links)	Organization chart, with roles; skill sets; security issues.	Business master schedule	Business plan
Information System Model	Data model (converged entities, fully normalized)	Essential Data flow diagram; application architecture	Distributed system architecture	Human interface architecture (roles, data, access)	Dependency diagram, entity life history (process structure)	Business rule model
Technology Model	Data architecture (tables and columns); map to legacy data	System design: structure chart, pseudo-code	System architecture (hardware, software types)	User interface (how the system will behave); security design	"Control flow" diagram (control structure)	Business rule design
Detailed Representation	Data design (denormalized), physical storage design	Detailed Program Design	Network architecture	Screens, security architecture (who can see what?)	Timing definitions	Rule specification in program logic
Function System	Converted data	Executable programs	Communications facilities	Trained people	Business events	Enforced rules

Example of
Zachman's
6x6
matrix,
top and left
sides are
main
views.

	WHAT	HOW	WHERE	WHO	WHEN	WHY	
SCOPE CONTEXTS	Inventory Identification  Inventory Types	Process Identification  Process Types	Network Identification  Network Types	Organization Identification  Organization Types	Timing Identification  Timing Types	Motivation Identification  Motivation Types	STRATEGISTS AS THEORISTS
BUSINESS CONCEPTS	Inventory Definition  Business Entity Business Relationship	Process Definition  Business Transform Business Input	Network Definition  Business Location Business Connection	Organization Definition  Business Role Business Work	Timing Definition  Business Cycle Business Moment	Motivation Definition  Business End Business Means	EXECUTIVE LEADERS AS OWNERS
SYSTEM LOGIC	Inventory Representation  System Entity System Relationship	Process Representation  System Transform System Input	Network Representation  System Location System Connection	Organization Representation  System Role System Work	Timing Representation  System Cycle System Moment	Motivation Representation  System End System Means	ARCHITECTS AS DESIGNERS
TECHNOLOGY PHYSICS	Inventory Specification  Technology Entity Technology Relationship	Process Specification  Technology Transform Technology Input	Network Specification  Technology Location Technology Connection	Organization Specification  Technology Role Technology Work	Timing Specification  Technology Cycle Technology Moment	Motivation Specification  Technology End Technology Means	ENGINEERS AS BUILDERS
COMPONENT ASSEMBLIES	Inventory Configuration  Component Entity Component Relationship	Process Configuration  Component Transform Component Input	Network Configuration  Component Location Component Connection	Organization Configuration  Component Role Component Work	Timing Configuration  Component Cycle Component Moment	Motivation Configuration  Component End Component Means	TECHNICIANS AS IMPLEMENTERS
OPERATIONS CLASSES	Inventory Instantiation  Operations Entity Operations Relationship	Process Instantiation  Operations Transform Operations Input	Network Instantiation  Operations Location Operations Connection	Organization Instantiation  Operations Role Operations Work	Timing Instantiation  Operations Cycle Operations Moment	Motivation Instantiation  Operations End Operations Means	WORKERS AS PARTICIPANTS
	INVENTORY SETS	PROCESS TRANSFORMATIONS	NETWORK NODES	ORGANIZATION GROUPS	TIMING PERIODS	MOTIVATION REASONS	

[<https://cdn.visual-paradigm.com/guide/enterprise-architecture/what-is-zachman-framework/01-zachman-framework.png>]

[softwarearchitectureinpractice/]

Definition: *The software architecture of a program or computing system is the structure or structures of the system, which comprise*

- **software elements,**
 - **the externally visible properties of those elements,**
 - **and the relationships among them.**
- "Externally visible" properties are those assumptions other elements can make of an element, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on.

Implications of this definition:

Architecture defines software elements

- **how the elements relate to each other**
- **an architecture is foremost an abstraction of a system that suppresses details of elements that do not affect how they use, are used by, relate to, or interact with other elements. (only public part)**

Systems comprise more than one structure

- **no one structure can claim to be the architecture**
- **Relationships and elements might be**
 - runtime related ("send data", processes)
 - nonruntime related ("submodel of", "inherits from", class)

Every computing system with software has a software architecture

- **it does not necessarily follow that the architecture is known to anyone.**
- **An architecture can exist independently of its description or specification**

The behavior of each element is part of the architecture

- **allows elements to interact with each other**
- **Add specification and properties to the elements and relationships**

The definition is indifferent as to whether the architecture for a system is a good one or a bad one

- **Evaluating the SW architecture in the context of usa.**

open system architecture

Vendor-independent, non-proprietary, computer system or device design based on official and/or popular standards. **It allows all vendors** (in competition with one another) **to create add-on** products that increase a system's (or device's) **flexibility, functionality, interoperability, potential use, and useful life**. And enables the users **to customize and extend** a system's (or device's) capabilities to suit individual requirements. Also called **open architecture**.

[<http://www.businessdictionary.com/definition/open-system-architecture.html>]

Highlights - What to remember

- software systems can be classified in many many ways
- there are also many architectural views, by which you can describe and define a business and software system
- framework is one realisation of an architecture
- At requirements and design phase when defining the architecture and framework, you have better to take possible future extensions into account, e.g. interfaces (API = application programming interface) to other systems or future extensions. Those are easier to plan right from the beginning, than add later.